**University of Central Punjab**

**Faculty of Information Technology**

**Data Structures and Algorithms**
**Spring 2024**

| Lab 04 | |
|---|---|
| **Topic** | • Abstract Classes<br>• Templates<br>• Stacks<br>• Stack Application<br>• Infix to prefix<br>• Infix to postfix |
| **Objective** | The basic purpose of this lab is to implement ADT of stack, Infix to Postfix conversion, infix to Prefix Conversion. |

**Instructions:**

- Indent your code.
- Comment your code.
- Use meaningful variable names.
- Plan your code carefully on a piece of paper before you implement it.
- Name of the program should be same as the task name. i.e. the first program should be Task_1.cpp
- **void main() is not allowed. Use int main()**
- **You have to work in multiple files. i.e separate .h and .cpp files**
- **You are not allowed to use system("pause")**
- **You are not allowed to use any built-in functions**
- **You are required to follow the naming conventions as follow:**
    - **Variables:** firstName; (no underscores allowed)
    - **Function:** getName(); (no underscores allowed)
    - **ClassName:** BankAccount (no underscores allowed)

**Students are required to complete the following tasks in lab timings.**

**Task 1**

Create a C++ generic abstract class named as **Stack**, with the following:

**Attributes:**

1. Type * stackArray;
2. int maxSize;
3. int stackTop;

**Functions:**

virtual void Push(Type) = 0;

- Should add the element at the top of

stack virtual Type Pop() = 0;

- Should remove the element from the top of stack

**Task 2**

Using the class made in task 1, make another derived class named as **InfixConverter**. From abstract class use the push and pop functions into the derived class accordingly for converting given Infix expression to Prefix/Postfix expressions. Make logic for converting Infix to Prefix and Postfix expressing through stack. The derived class **InfixConverter** should have following functionalities:

**bool empty():** Returns whether the **Stack** is empty or not.

**bool full():** Returns whether the **Stack** is full or not.

**Int size():** Returns the current size of Stack.

**Type top():** Returns the last element of the **Stack.**

**void infixToPostfix():** Should convert given infix expression to postfix expression.

***INPUT:*** (A + B) * C - D

***OUTPUT:*** AB + C * D -

**void infixToPrefix():** Should convert given infix expression to prefix expression.

***INPUT:*** (A + B) * C - D

***OUTPUT:*** - * + ABCD

**void display():** Should display the stack.

**Implement both pure virtual functions Push () and pop() declared in base in InfixConverter**

After Implementation of the functions in **InfixConverter** create menu based program to perform the following operations:

1. Press 1 to add a new item to the stack. **void push(Type)**
2. Press 2 to remove and return the last element from the stack. **Type pop()**
3. Press 3 to check if stack is full. **bool full()**
4. Press 4 to check if stack is empty. **bool empty()**
5. Press 5 to return the size of stack. **int size()**

6. Press 6 to return the top/last element of stack. **int size()**
7. Press 7 to convert infix expression to postfix expression. **void infixToPostfix(string s)**
8. Press 8 to convert infix expression to prefix expression. **void infixToPrefix(string s)**
9. Press 9 to display stack. **void display()**
10. Press 0 to exit.

- Write non-parameterized constructor for the above class.
- Write Copy constructor for the above class.
- Write Destructor for the above class.